Europäisches Patentamt | European Patent Office | Office européen des brevets

REC'D 26 NOV 2003
WIPO PCT

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

03009906.3

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

CERTIFIED COPY OF PRIORITY DOCUMENT

BEST AVAILABLE COPY

Anmeldung Nr:
Application no.: 03009906.3
Demande no:

Anmeldetag:
Date of filing: 30.04.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

PACT XPP Technologies AG
Muthmannstrasse 1
80939 München
ALLEMAGNE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F15/76

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PT RO SE SI SK TR LI

The present invention relates to a method for data processing using a reconfigurable array.

In data processing using a reconfigurable array there may arise situations wherein a higher or lower load for as system may be given, depending e.g. on the array size and/or the load at run time due to the number of tasks to be executed during run-time. It would be preferred to allow for an optimised data processing in such situations.

The present invention aims at improving data processing using reconfigurable arrays.

Some of the important aspects of the present invention can be derived from the claims. Other important aspects will be obvious from the description and the drawings.

Accordingly, the present invention suggests inter alia to expand a binary code for a function fully while de-expanding the binary code at run time and or prior to execution, e.g. during program installation on a system so as to allow for a partially sequential way of processing of data on an array which is adapted to allow at least for a certain mininimum degree of sequentiality by allowing for the implementation of sequencers eg in ways known from previous applications of the present applicant and/or inventor.

Accordingly, one method of implementing the invention is to determine at compile time from the high level language code a structure having a high, preferably an optimum level of paral-

lelism and to then determine at run time whether a configuration can be executed using said optimum level of parallelism or whether, for example due to current load conditions of the system and /or due to system restrictions, a configuration needs to be executed wherein said parallelism of said structure is somewhat reduced by executing a somewhat "folded" configuration on sequentially operating parts of the array. These configurations can be regarded as being folded since they are not loaded onto the maximum array space that the program structure allows for, but are restricted in their size by executing certain of the operations one after the other on the same array element, thus folding said configurations that would otherwise be executed at different locations onto each other.

Claims

1. Method for data processing using a reconfigurable array of logical elements
wherein a configuration for execution of a certain data processing function is determined and loaded onto the arrray for execution
characterised in that
code representative for said data processing function is generated, said generated code is modified according to certain conditions met prior to execution and a configuration is loaded onto said array according to said modified code.

2. Method according to claim 1 wherein the modification of said code comprises a determination of a suitable bus structure and or -organsiation for said array.

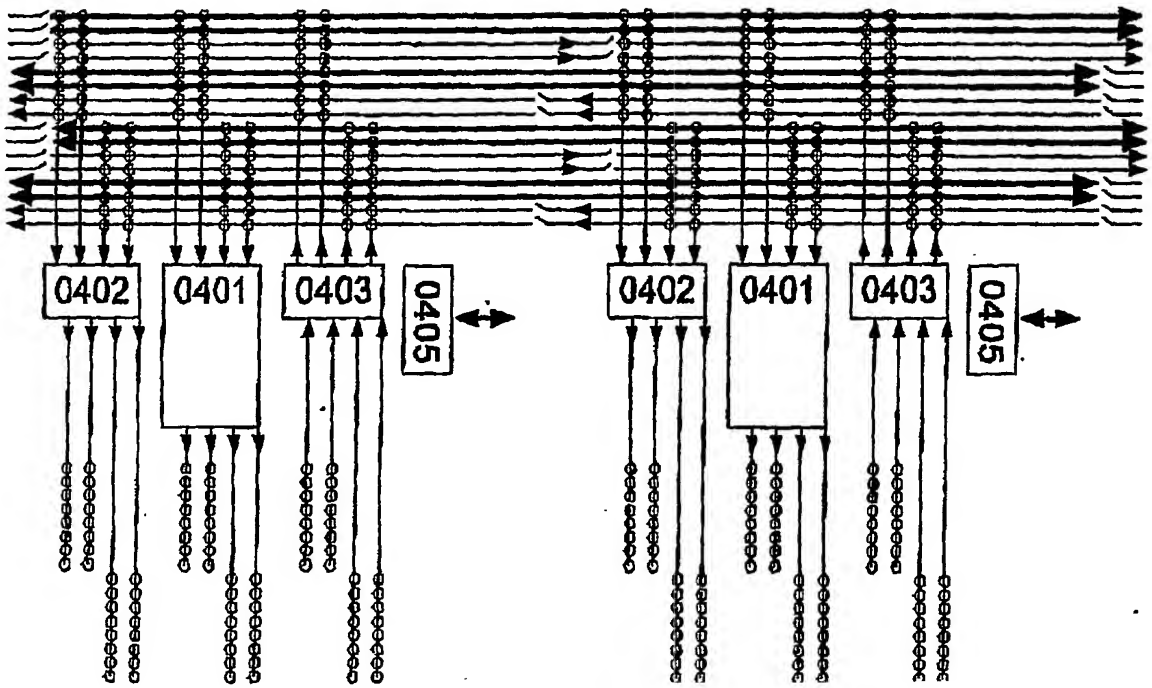
3. Method according to claim 2, wherein said bus structure is determined during run time, in particular by a place- and/or route module, said place-and/or route being in particular comprised in or called by a configuration loading module such as a program-loader.

4. Method according to claim 2, wherein said bus structure is determined according to a given system by selecting a preferred one among certain predetermined codes modified relative to one another at compile time.

5. Method according to any previous claim, wherein code is modified so as to have a different degree of sequentiality.

Interconnect dominated (Basics)

- Use of (staggered) „long tracks“
  - Increases operating frequency
  - Decreases transistor count and area
  - Optimizes metalization layers
  - Decreases power dissipation



Interconnect dominated
(Frequency problem at the market place)

2

Three additional strategies:

- Operate at different frequencies, i.e.:
  - MIPS µP Core at 400 MHz
  - PACT XPP Core at 200 MHz
- Registers in each bus-connect
- PAEs act sequentially, one output each 2nd or 4th clock cycle

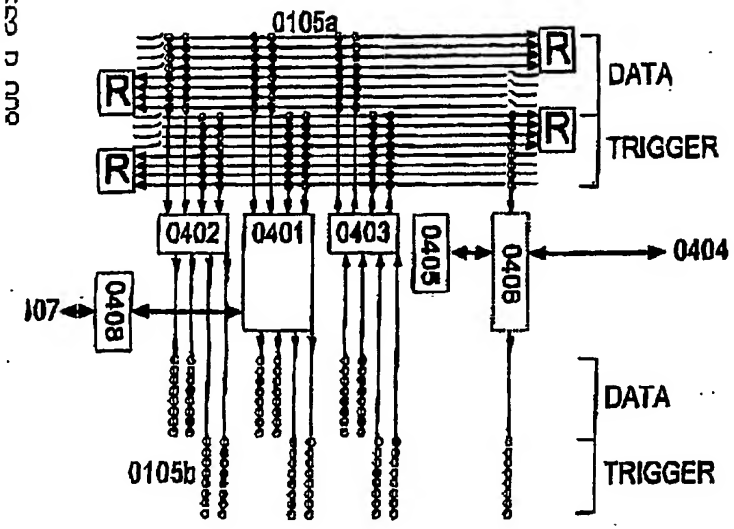


Fig. 4

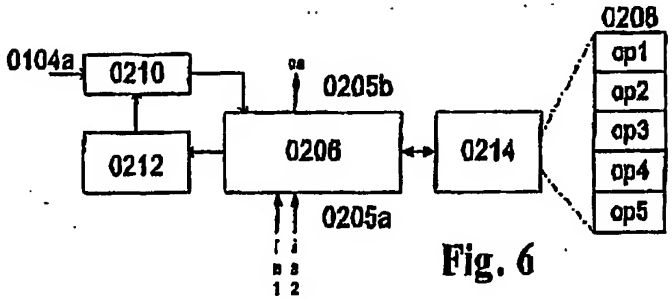


Fig. 6

Trade-off between area and sequentiallity

- Basic operation of compiler technology can be used to achieve abstraction layer
- Generate „compressed" configurations, which are expanded on larger arrays while loading
- No minimum array but maximum array defined by compiler setting
  - Tradeoff: Number of (re-)configurations vs. usable ALU-PAEs

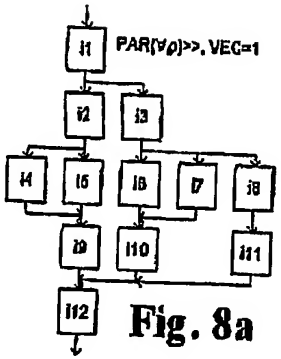
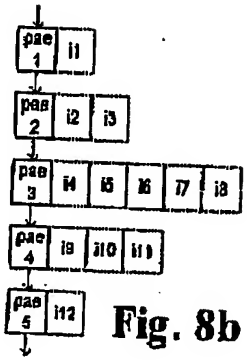


Fig. 8a



VEC = 1
cy1 = 1
cy2 = 1
cy3 = 1
cy4 = 1
cy5 = 1

Fig. 8b

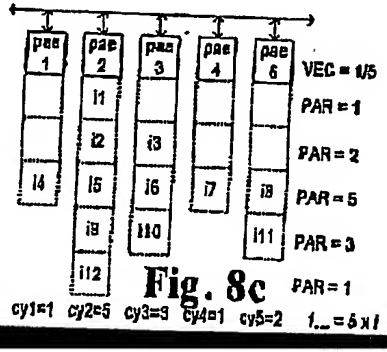
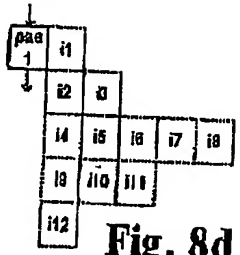


Fig. 8c



VEC=1/5 PAR(∀p)=1
cy1 = 5

Fig. 8d

❑ Basic Operation Method

- LOAD/STORE processor
- RAM-PAEs act as Vector-Registers (2D/3D)
- Irregular data access patterns are linearized by LOAD/STORE while accessing RAM-PAEs
  - Can be done by µP also!
- LOAD ... $Conf_1$ ... $Conf_2$ ... ... $Conf_n$... STORE
- Each Configuration is regarded as an OpCode
- No Configuration/Array internal status

❑ Code Analysis

- Data Dependency Analysis
- Data Flow Analysis
- Interprocedural Alias Analysis
  - Pointer analysis: statically allocated data, dynamically allocated data
- Interprocedural Value Range Analysis

Code Optimizations

- Loop Transformations
  - Loop Normalization
  - Loop Reversal
  - Loop-Invariant Code Motion
  - Loop Unswitching
  - Loop Interchange
  - Loop Tiling
  - Loop Skewing
  - Loop Coalescing/Collapsing
  - Loop Fusion
  - Loop Distribution
  - Loop Unrolling
  - Loop Peeling
  - Loop Splitting
  - Loop Pushing/Embedding
  - Strength Reduction
  - Induction Variable Elemination
  - Strip Mining
  - Scalar Expansion
  - Array Contracting/Shrinking
  - Scalar Replacement
  - Reduction Recognition
  - Idiom Recognition
  - Procedure Inlining
  - Software Pipelining
  - Vector Statement Generation
  - Node Splitting
  - If Conversion
  - Statement Reordering

Combined caches and RAM-PAEs

- RAM-PAEs RAM is „embedded" into cache
- RAM-PAEs can operate like cache-lines
  - Homogeneous embedded in cache
  - Handling access rights between µP and XPP
  - Handling context switching / hyperthreading
  - Abstracting non linear address patterns

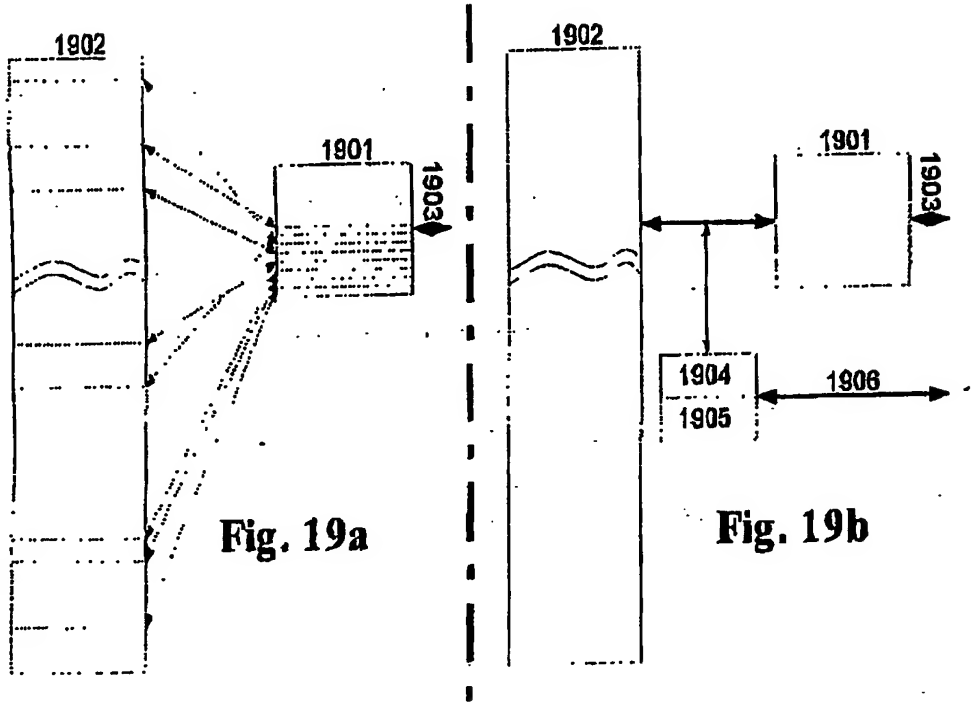


Fig. 19a

Fig. 19b

Multithreading

7

- XPP operates like an RISC-Processor
- RAM-PAEs act like registers
- Each configuration is atomar (unbreakable)
- Configurations running time is limited

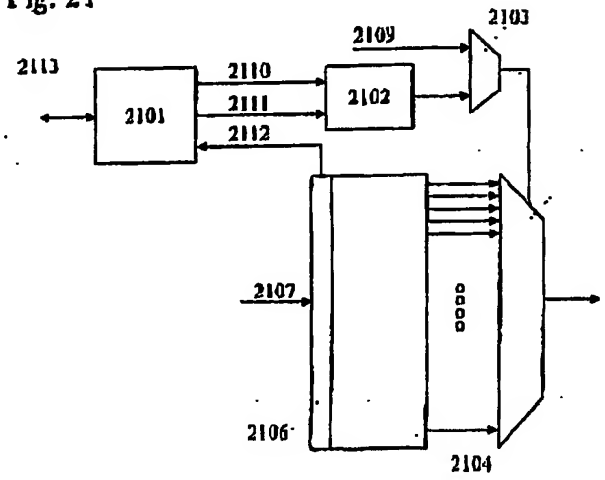
- LOAD Configuration
  - Loads external data into internal RAM-PAEs
- Data operations (one or multiple configurations)
  - Unbreakable – no internal status to be saved!
- STORE Configuration
  - Stores internal data into external RAM-PAEs

- Interrupts (Task/Thread-Switches) only between (re)configurations not at runtime

XPP Technology allows sequential processing

- Within ALU-PAEs using the configuration register file as a random access code memory
- Coupling an ALU-PAE with a RAM-PAE. ALU-PAE acts like a µC, RAM-PAE is according Data- and Code-Memory
  - As an enhancement IO-PAEs can be used to access peripherials and external memory

Fig. 21



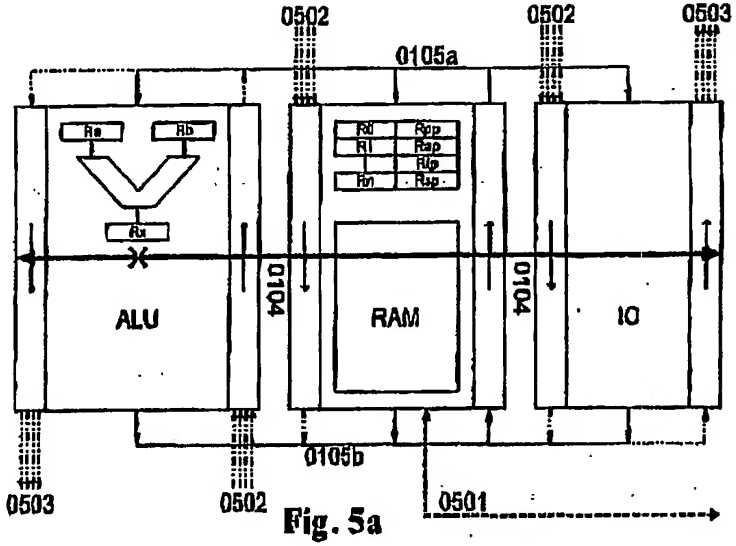
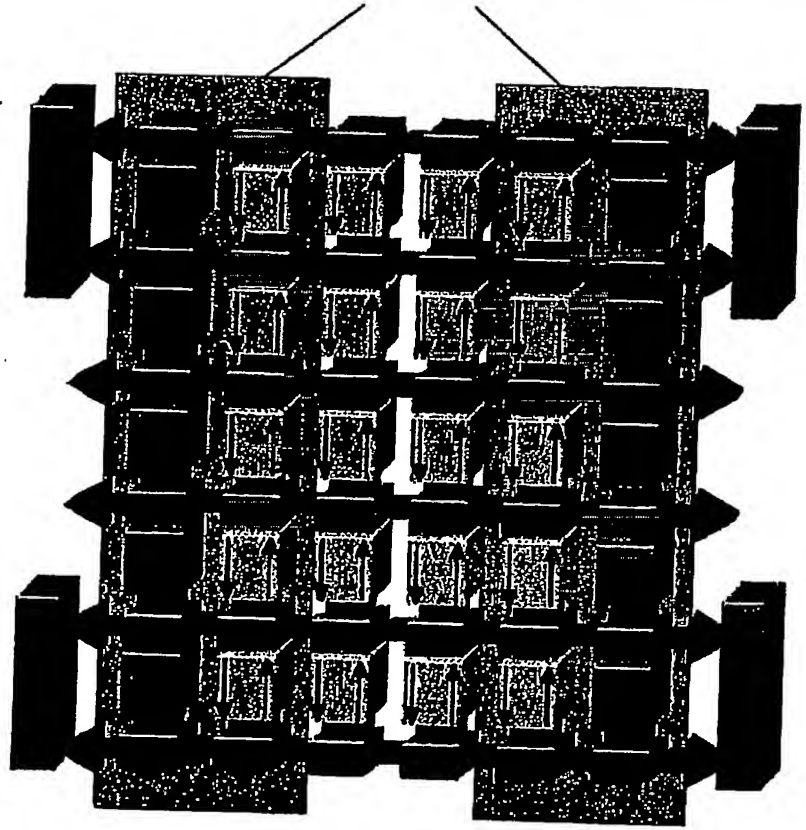


Fig. 5a

- Optimum trade-off between sequencing and dataflow processing

Configurable ALU-PAE / RAM-PAE Sequencers



Complex operations like 40-bit, Floating Point

- Handled by sequential processing within PAEs
  - I.e. Floating-Point, Division etc can be emulated by sequential multicycle PAE operations
  - Higher precision is calculated as a multicycle operation
    - results are transfered in two bus cycles